Übungen zum Kapitel 3

Datenstrukturen und Funktionen in Python

Erstellt und überarbeitet: armin.baenziger@zhaw.ch, 9. Januar 2020

In [1]:
```
%autosave 0
```

```
Autosave disabled
```

Für die ersten Fragen erzeugen wir zuerst ein Tupel:

In [2]:
```
tup1 = (-1, 3, -5, 9, 1)
```

(A.1) Weisen Sie das erste Element in `tup1` der Variable `a` zu.

In [3]:
```
a = tup1[0]
a
```

Out[3]:
```
-1
```

(A.2) Prüfen Sie, ob das erste Element in `tup1` positiv ist.

In [4]:
```
a > 0
```

Out[4]:
```
False
```

(A.3) Erstellen Sie die Liste `list1` aus `tup1` .

In [5]:
```
list1 = list(tup1)
list1
```

Out[5]:
```
[-1, 3, -5, 9, 1]
```

(A.4) Hängen Sie der Liste `list1` den Wert `0` an. Tipp: `append` -Methode

In [6]:
```
list1.append(0)
list1
```

Out[6]:
```
[-1, 3, -5, 9, 1, 0]
```

(A.5) Löschen Sie den soeben hinzugefügen Wert wieder mit der Methode `pop()` . Hinweis: `pop` löscht automatisch den letzten Wert einer Liste, wenn man kein Argument übergibt.

In [7]:
```
list1.pop()
list1
```

Out[7]:
```
[-1, 3, -5, 9, 1]
```

(B.1) Erzeugen Sie folgende Liste mit der `range` -Funktion: `list2 = [0, 5, 10, 15, 20, 25, 30]` .

In [8]:
```
list2 = list(range(0, 31, 5))
list2
```

Out[8]:
```
[0, 5, 10, 15, 20, 25, 30]
```

(B.2) Überprüfen Sie, ob `35` in der Liste `list2` vorkommt.

In [9]:
```
35 in list2
```

Out[9]:
```
False
```

(B.3) Erstellen Sie mit einem `for` -Loop die neue Liste `list3` , welche alle *positiven* Werte aus `list1` enthält.

In [10]:
```
list3 = []  # Leere Liste initialisieren. Mit append() ergänzen.

for x in list1:
    if x > 0:
        list3.append(x)

list3
```

Out[10]:
```
[3, 9, 1]
```

(B.4) Lösen Sie **(B.3)** nun mit einer List-Comprehension.

In [11]:
```
list3 = [x for x in list1 if x > 0]
list3
```

Out[11]:
```
[3, 9, 1]
```

(B.5) Generieren Sie `list4` mit den *sortieren* Werten aus `list1` . Achten Sie darauf, dass `list1` *nicht* verändert wird. Hinweis: Verwenden Sie nicht die Methode `sort` , sondern die Funktion `sorted` .

In [12]:
```
list4 = sorted(list1)
list4
```

Out[12]:
```
[-5, -1, 1, 3, 9]
```

(C.1) Entnehmen Sie aus `list1` die ersten zwei Elemente (*slicing*).

In [13]:
```
list1[:2]
```

Out[13]:
```
[-1, 3]
```

(C.2) Entnehmen Sie aus `list1` die letzten zwei Elemente.

In [14]:
```
list1[-2:]
```

Out[14]:
```
[9, 1]
```

(C.3) Entnehmen Sie aus `list4` jedes zweite Element. Tipp: Zweimal Doppelpunkt verwenden.

In [15]:
```
list4[::2]
```

Out[15]:
```
[-5, 1, 9]
```

Für die nächsten Fragen erzeugen wir zuerst ein Dict:

In [16]:
```
dict1 = {'A': 'Alice', 'B': 'Bert', 'C': 'Claudia'}
```

(D.1) Lesen Sie aus `dict1` den Namen unter Schlüssel "C" aus.

In [17]:
```
dict1['C']
```

Out[17]:
```
'Claudia'
```

(D.2) Fügen Sie `dict1` den Vornamen "Wes" mit Schlüssel "W" hinzu.

In [18]:
```
dict1['W'] = 'Wes'
dict1
```

Out[18]:
```
{'A': 'Alice', 'B': 'Bert', 'C': 'Claudia', 'W': 'Wes'}
```

(D.3) Löschen Sie den Schlüssel "W" mit entsprechendem Wert wieder aus `dict1` mit `del` .

In [19]:
```
del dict1['W']    # oder: dict1.pop('W')
dict1
```

Out[19]:
```
{'A': 'Alice', 'B': 'Bert', 'C': 'Claudia'}
```

(E.1) Erstellen Sie das Set `set1` aus der Liste `list1` , die wir vorher schon erstellt haben.

In [20]:
```
set1 = set(list1)
set1
```

Out[20]:
```
{-5, -1, 1, 3, 9}
```

(E.2) Erstellen Sie `set2` mit den Elementen B, D und E.

In [21]:
```
set2 = {'B', 'D', 'E'}
set2
```

Out[21]:
```
{'B', 'D', 'E'}
```

(E.3) Verwenden Sie die Funktionen `set` und `len` um die Anzahl *unterschiedlicher* Zeichen in `list5` zu ermitteln.

In [22]:
```
list5 = list('Donaudampfschifffahrt')
len(set(list5))
```

Out[22]:
```
15
```

(F.1) Erzeugen Sie mit einer "List-Comprehension" die Folge `[0, 1, 4, 9, 16, 25, 36]`, also i^2 für $i = 0, 1, \ldots, 6$.

In [23]:
```
[x ** 2 for x in range(7)]
```

Out[23]:
```
[0, 1, 4, 9, 16, 25, 36]
```

(G.1) Erstellen Sie die Funktion `quadriere`, welche ein Argument übernimmt, welches quadriert zurückgegeben wird.

In [24]:
```
def quadriere(x):
    return x ** 2

# Beispiel:
quadriere(4)
```

Out[24]:
```
16
```

(G.2) Erstellen Sie die Funktion `betrag()`, welche den Betrag einer beliebigen Zahl zurückgibt.

In [25]:
```
def betrag(x):
    if x >= 0:
        return x
    else:
        return -x

betrag(-5)
```

Out[25]:
```
5
```

Ende der Übung